8. Rearrange

Program Name: Rearrange.java Input File: rearrange.dat

Reverse Polish notation (RPN) calculators are considered to be optimal calculators to use because they never require parentheses. This is done by using a stack-based approach, where each operator occurs after all of its operands. You will be converting from the standard infix form of equations to this described postfix form. This involves rearranging the operators and operands so that they evaluate correctly without thinking of operator precedence. Your algorithm will only have to work with + - * /.

An advantage of RPN is that it obviates the need for parentheses that are required by infix. While "3 - 4 * 5" can also be written "3 - (4 * 5)", that means something quite different from "(3 - 4) * 5". In postfix, the former could be written "3 4 5 * -", which unambiguously means "3 (4 5 *) -" which reduces to "3 20 -"; the latter could be written "3 4 - 5 *" (or 5 3 4 - *, if keeping similar formatting), which unambiguously means "(3 4 -) 5 *".

One example of the desired conversion is “3 + 4” :: “3 4 +”. Another, more complicated example is “3 + 2 * 5” :: “3 2 5 * +”.

Input
The first line contains a single integer, T, which states the number of test cases to follow. Each test case will consist of a single line that contains a function in infix form, using only integers. There will be no parentheses, and the usual Java operator precedence rules apply.

Constraints
$0 < T < 30$
The number of operands < 100
The number of operators < 100
$0 <$ each number < 1000

Output
For each test case, print out on its own line the function in postfix form. All numbers must be given in the same order as given, but the order of the operators may vary from their order in the infix form. There is a unique solution for each problem. You should output a single space between each number/operator.

Example Input File
```
2
4 * 2 + 1
3 + 1 / 7
```

Example Output to Screen
```
4 2 * 1 +
3 1 7 / +
```